This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2005 proceedings.

ServerCast: Efficient Cooperative Bulk Data Distribution Scheme for Content Distribution Networks

Tin-Man T. Kwan, Kwan L. Yeung
Department of Electrical and Electronic Engineering
The University of Hong Kong Hong Kong, PRC
Fax: (852) 2857-8493 Fax: (852) 2559-8738
E-mail: {tmtkwan, kyeung}@eee.hku.hk

***Abstract* — We study bulk data distribution schemes for content distribution networks (CDNs) using application-level overlay. Given the outbound bandwidth of all the CDN edge servers, a fluid-flow based analytical model is constructed to derive the optimal bandwidth allocations at the origin server and among all the edge servers. From which, a lower bound on the content update time is obtained. In order to have an efficient and practical scheme for bulk data distribution, ServerCast is designed to take advantages of the analytical results. Simulations are conducted to evaluate the performance of ServerCast. We show that ServerCast not only outperforms the two existing schemes, FastReplica and Multiple Unicast, but also yields a content update time within 18% of the derived lower bound.**

***Index Terms* -- Content Distribution Network, ServerCast, Application-level Overlay**

I. INTRODUCTION

In the past decade, with the exponential growth of the Internet, a single web server at a single point of presence could no longer satisfy the demand of the fast growing number of users. Content Distribution Networks (CDNs) such as Akamai [1] are then deployed to offer load-alleviating hosting services to web content providers. CDN replicates customers' content in their edge servers. When content providers update their contents, CDN updates its edge servers. There are three popular methods [2] for CDN to update its edge servers (also known as content distribution/replication) in the Internet: satellite distribution, IP multicast distribution and application-level overlay distribution. The satellite distribution requires installation of special hardware and the supporting infrastructure and services, which are in general quite expensive. Meanwhile, IP multicast distribution relies on IP multicast support in routers, which is not widely deployed across the Internet yet, especially in lower-tier ISPs. Application-level overlay distribution overcomes the deployment hurdle faced by IP multicast by requiring no changes to existing routers. Instead, they construct overlay topologies using unicast connections.

Application-level overlay has the advantages of easy implementation, low cost and congestion alleviation (by rerouting around congested areas of the Internet) [3]. But due to the capacity limit at the origin server (i.e. the server where the content updates first arrive) and low efficiency in duplicating data transmission (as compared with IP multicast), it usually takes a long time to update all edge servers. On the other hand, with application-level overlay for content distribution, edge servers are explicitly required to cooperate. If edge servers can actively collaborate in an informed manner, the content distribution performance can be significantly improved.

A promising approach is to let the edge servers contribute their bandwidth resources to help each other. Consider downloading a large file from a single origin server to all the edge servers. The origin server can partition the file into n blocks and uploads blocks to different edge servers. The edge servers can collaborate with each other to assemble all the n blocks to reconstruct the original file. In order to minimize the total time required for the *last* edge server to receive the entire file, or *content update time*, different data distribution schemes can be used [4-6]. Please refer to the next section for a review on some related schemes.

In this paper, a novel bandwidth allocation scheme (ServerCast) is proposed to minimize the content update time. Unlike FastReplica in [4], our ServerCast allows both the origin server and the edge servers transmit data simultaneously. In this regard, ServerCast is similar to BitTorrent [6]. However, the highly dynamic nature of individual peers (e.g. unpredictable independent join and leave of peers, flash crowds and different starting times) on peer-to-peer network is remarkably different from the "always-up" CDN edge servers. Hence, for CDN content updates, the whole transmission process is relatively predictable.

Given the outbound bandwidth of all the edge servers (including the origin server), a fluid-flow based analytical model is constructed to derive the optimal bandwidth allocations at the origin server and among all the edge servers. From which, a lower bound on the content update time can be obtained. In order to have an efficient and practical scheme for bulk data distribution, ServerCast is designed based on the analytical model. Simulations are conducted to evaluate the performance of ServerCast. We show that ServerCast not only outperforms the two existing schemes, FastReplica [4] and Multiple Unicast, but also yields a content update time within 18% of the theoretical lower bound.

The rest of the paper is organized as follows. Section II briefly describes the related work. In Section III, we construct the analytical model for bandwidth allocation. Based on it, our proposed bulk data distribution scheme ServerCast is presented in Section IV. Performance evaluation is conducted in Section V and conclusion is given in Section VI.

II. RELATED WORK

This work is supported by Hong Kong Research Grant Council Earmarked Grant HKU 7150/04E.

Existing research on CDNs and server replication primarily focuses on either the techniques for efficient redirection of user requests to appropriate servers, or content servers placement strategies for reducing the latency of end-users [7-11]. Recently, some interesting proposals on bulk data distribution are reported. They can be divided into two categories: solutions for CDN bulk transfer, and solutions for peer-to-peer networks.

A. CDN Bulk Transfer Solutions

FastReplica [4] also aims at minimizing the content update time. At the origin server, the original file is first partitioned into n equal size blocks (where n is the number of edge servers) and each block is transferred to a different edge server in the *distribution phase*. In the *collection phase* followed, each edge server propagates its block to all the remaining *n*-1 edge servers. In so doing, each edge server takes the advantages of *n*-1 parallel transmission paths. Comparing with sequential unicast and multiple unicasts, a significant reduction in content update time can be achieved. However, separating the file transfer into two (non-overlapped) phases can lead to underutilization of the edge servers' outbound bandwidth in the distribution phase, and the origin server's outbound bandwidth in the collection phase.

B. Peer-to-Peer Solutions

Among various bulk data distribution schemes for peer-to-peer networks, BitTorrent and Slurpie have attracted a lot of attentions. But due to their primary peer-to-peer nature, such as up-and-down dynamics and frequency, file source location, etc, optimization constraints for CDNs are quite different from peer-to-peer networks.

1) *BitTorrent*

Since peers' activities are dynamic and unpredictable, BitTorrent [6] does not implement any mechanisms to centrally allocate outbound bandwidth. Instead, it implements "chocking" mechanism which is used to stop a sender from sending blocks. The primary aim of this is to encourage "tit-for-tat" cooperation rather than reducing the total distribution time (i.e. content update time).

2) *Slurpie*

Slurpie [5] aims at offloading the web clients' inbound traffic from a single web server. Similar to BitTorrent and our ServerCast, it also utilizes the outbound bandwidth of the receivers to cooperatively assist the distributions. They have implemented a continuous coarse bandwidth estimation (throttle-back, underutilized and at-capacity) to decide the addition of peers in *mesh formation* and *update propagation*.

Compared with our ServerCast scheme, Slurpie tries to fully utilize the link bandwidth by adding peers as long as the link is under-utilized during the distribution process. In ServerCast, as detailed in the subsequent sections, we use a more precise central bandwidth allocation scheme (via exchanging collaborative information).

III. Analytical Model for Bandwidth Allocation

A. Assumptions

We consider a typical CDN network as shown in Fig. 1. Without loss of generality, we assume that the bottleneck in a content distribution system is the outbound bandwidth of each edge server (including the origin server). In other words, we assume that the total rate by which an edge server can

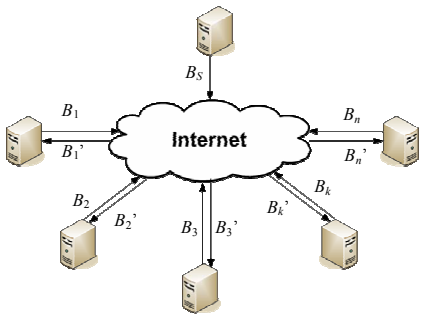

Fig. 1 CDN Network

contribute data to all other edge servers is limited by its outbound bandwidth B_i. (For the origin server, we denote its outbound bandwidth by B_s.) The total rate by which an edge server can receive information from all other edge servers (i.e. its inbound bandwidth) is assumed to be always sufficient, and is denoted by B_i'. We also assume that the inbound and the outbound bandwidths of a server are independent.

To justify the above assumptions, we consider the following arguments. Based on the concept of moving contents closer to the clients, CDN edge servers are usually located in the lower-tier ISPs. In general, we believe that the outbound bandwidth of an edge server for CDN updates should be limited in order to constrain the amount of traffic originating to the upper-tier ISPs, as it usually incurs higher cost. (This is supported by the general customer-provider relationship between the lower-tier and upper-tier ISPs.) Meanwhile, an edge server should reserve sufficient inbound bandwidth for its own content updates. So in general an edge server will dedicate more bandwidth to inbound direction than the outbound. Notably that similar assumptions are adopted in, e.g. FastReplica.

Suppose there are n edge servers (n+1 if the origin server is included) in a CDN. Edge server k equally[1] allocates its outbound bandwidth to the remaining n-1 edge servers, i.e. $1/(n-1)B_k$ each. This capacity is used to forward a copy of the data it received from the origin server to every other edge server. Let $B_{s,k}$ be the outbound bandwidth of the origin server allocated to edge server k. Then the aggregated server outbound bandwidth allocations must be less than or equal to the total outbound bandwidth of the origin server, i.e. $B_s \geq \sum_{i=1}^{n} B_{s,i}$.

As the propagation and processing delays are usually small comparing with the time required to complete the bulk data transfer, they are therefore ignored in our analysis below. Besides, we adopt the fluid-flow model for data transfer, which implies that data can be infinitesimally small.

B. Analytical Model

By the continuity of fluid model, the actual outbound data

[1] From the subsequent analysis in Section III, we can see that *equal* outbound bandwidth allocation among edge servers will not increase the content update time. In fact, for a given origin server, the content update time depends purely on how the origin server allocates its outbound bandwidth to *balance* the total inbound bandwidth received by each edge server. Equal allocation is assumed here because it is simple to implement.

rate contributed by edge server k to any other edge server i ($i \neq k$) is $\min\{B_k/(n-1), B_{s,k}\}$. Consider server k, its total inbound data rate consists of two components: the contribution from the origin server ($B_{s,k}$) and the contributions from all other edge servers, or

$$B_{s,k} + \sum_{\substack{i=1 \\ i\neq k}}^{n} \min\left\{\frac{B_i}{n-1}, B_{s,i}\right\}$$

The amount of time required by edge server k to retrieve the whole file with file size F is thus given by:

$$T_k = \frac{F}{B_{s,k} + \sum_{\substack{i=1 \\ i\neq k}}^{n} \min\left\{\frac{B_i}{n-1}, B_{s,i}\right\}} = \frac{F}{B_{s,k} + \sum_{i=1}^{n} \min\left\{\frac{B_i}{n-1}, B_{s,i}\right\} - \min\left\{\frac{B_k}{n-1}, B_{s,k}\right\}} \quad (1)$$

Intuitively, if every edge server uses the same amount of time to retrieve the file, then the overall content update time can be minimized. From (1), we can see the differences among different T_k's solely depends on the value of $B_{s,k} - \min\{B_k/(n-1), B_{s,k}\}$. If for all k, we can have:

$$B_{s,k} - \min\left\{\frac{B_k}{n-1}, B_{s,k}\right\} = \text{constant} \quad (2)$$

Then the minimum content update time can be achieved. In fact, (2) represents the difference between the allocated bandwidth from the origin server and the outbound bandwidth to other servers, which must be positive.

Assume $B_{s,k} \geq B_k/(n-1)$ for all k. From (2), we have:

$$B_{s,k} - \frac{B_k}{n-1} = \text{constant} \quad (3)$$

$B_{s,k} - B_k/(n-1)$ represents the excess outbound bandwidth allocated to an edge server. Then the total excess outbound bandwidth allocated to all edge servers is $B_s - (\sum_{i=1}^{n} B_i)/(n-1)$. If the total amount of excess bandwidth is equally allocated to all the servers, each server will get an extra bandwidth of

$$B_{s,k} - \frac{B_k}{n-1} = \frac{1}{n}\left(B_s - \frac{1}{n-1}\sum_{i=1}^{n} B_i\right) = \text{constant}$$

This satisfies (3). Therefore, if $B_{s,k} \geq B_k/(n-1)$ for all k, minimum content update time can be obtained with the following bandwidth allocation:

$$B_{s,k} = \frac{B_k}{n-1} + \frac{1}{n}\left(B_s - \frac{1}{n-1}\sum_{i=1}^{n} B_i\right) \quad (4)$$

Interestingly, the same conclusion in (4) holds for the case that Bs,k < Bk/(n–1) for all k. The only difference is that $[B_s - (\sum_{i=1}^{n} B_i)/(n-1)]/n$ in (4) becomes *negative*, which indicates the amount of (equal) bandwidth *reduction* required on $B_k/(n-1)$. In this case, the origin server does not have enough outbound capacity to fully utilize all the inter-edge server capacities. But the overall content update time can still be minimized with the bandwidth allocation in (4).

Theorem 1. If $B_{s,k} = B_k/(n-1) + [B_s - (\sum_{i=1}^{n} B_i)/(n-1)]/n$ is allocated to edge server k, then the minimum content update time is:

$$T = \begin{cases} \dfrac{nF}{B_1 + B_2 + B_3 + .. + B_n + B_s} & for \quad B_s \geq \dfrac{1}{n-1}\sum_{i=1}^{n} B_i \\ \dfrac{F}{B_s} & for \quad B_s < \dfrac{1}{n-1}\sum_{i=1}^{n} B_i \end{cases}$$

Proof: From (1), the minimum content update time is:

$$T = \max\{T_k\} = \max_{k\in\{1,\ldots,n\}}\left\{\frac{F}{B_{s,k} + \sum_{i=1}^{n}\min\left\{\frac{B_i}{n-1}, B_{s,i}\right\} - \min\left\{\frac{B_k}{n-1}, B_{s,k}\right\}}\right\}$$
$$= \frac{F}{\min_{k\in\{1,\ldots,n\}}\left\{B_{s,k} + \sum_{i=1}^{n}\min\left\{\frac{B_i}{n-1}, B_{s,i}\right\} - \min\left\{\frac{B_k}{n-1}, B_{s,k}\right\}\right\}} \quad (5)$$

If $B_s \geq \left(\sum_{i=1}^{n} B_i\right)/(n-1)$, substitute $B_{s,k} = B_k/(n-1) + [B_s - (\sum_{i=1}^{n} B_i)/(n-1)]/n$ into (5), we get

$$T = \frac{F}{\min_{k\in\{1,\ldots,n\}}\left\{\dfrac{\sum_{i=1}^{n} B_i}{n-1} + \dfrac{B_s - \dfrac{1}{n-1}\sum_{i=1}^{n} B_i}{n}\right\}} = \frac{nF}{B_1 + B_2 + B_3 + .. + B_n + B_s} \quad (6)$$

If $B_{s,k} = B_k/(n-1) + [B_s - (\sum_{i=1}^{n} B_i)/(n-1)]/n$ is *not* allocated to edge server k for some k, say

$$B_{s,k} = B_k/(n-1) + [B_s - (\sum_{i=1}^{n} B_i)/(n-1)]/n + \delta.$$

Then there exists at least one edge server with allocated bandwidth smaller than $B_{s,k} = B_k/(n-1) + [B_s - (\sum_{i=1}^{n} B_i)/(n-1)]/n$, say $B_{s,j} = B_j/(n-1) + [B_s - (\sum_{i=1}^{n} B_i)/(n-1)]/n - \delta''$. The content update time required in this case becomes $nF/(B_1 + B_2 + B_3 + .. + B_n + B_s - \delta'')$, which is larger than $nF/(B_1 + B_2 + B_3 + .. + B_n + B_s)$.

For the case of $B_s < \left(\sum_{i=1}^{n} B_i\right)/(n-1)$, by substituting $B_{s,k} = B_k/(n-1) + [B_s - (\sum_{i=1}^{n} B_i)/(n-1)]/n$ into (5), we get

$$T = \frac{F}{\min_{k\in\{1,\ldots,n\}}\{B_{s,1} + B_{s,3} + B_{s,2} + \ldots + B_{s,n}\}} = \frac{F}{B_s}.$$

The above T is minimized as B_s is the bottleneck in this case. This completes our proof for Theorem 1.

IV. SERVERCAST FOR BULK DATA DISTRIBUTION

In this section, we propose a practical bulk data distribution scheme (ServerCast) based on the analysis in the previous section. ServerCast consists of four steps. Step 1 allows all servers to exchange their outbound bandwidth information2 before the bulk data transfer takes place. The collected information is used to calculate the theoretical bandwidth allocation $B_{s,k}$ according to equation (4). Step 2 prepares the original file for transmission. In particular, it segments the file into blocks and then assembles the blocks into n groups, as shown in Fig. 2. Step 3 and Step 4 are operated in parallel. Step 3 governs the transmission at the origin server, where the blocks in Group k are sent to edge server k at the pre-calculated rate $B_{s,k}$, for $k = 1, \ldots, n$. Step 4 takes place at

[2] In real implementations, we can use, e.g. periodic bandwidth probing techniques [12,13] to measure the available bandwidth. In fact, the origin server could dynamically adjust its outbound bandwidth allocation so as to adapt to the varying traffic load in the Internet.

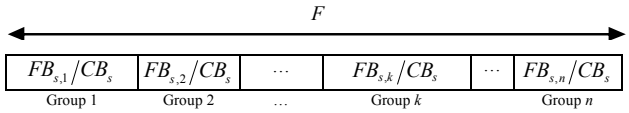

Fig. 2 File partitioning and block grouping

1. Exchange outbound bandwidth information among all the servers. Based on the collected statistics, calculate the idealized bandwidth allocation
$$B_{s,k} = B_k/(n-1)+[B_s-(\sum_{i=1}^{n} B_i)/(n-1)]/n.$$
2. Segment the original file F into F/C blocks, where C is the block size. Assemble the blocks into n groups: Group 1 has the first $(F \cdot B_{s,1})/(C \cdot B_s)$ blocks, Group 2 has the next $FB_{s,2}/CB_s$ blocks, …, Group n has the last $FB_{s,n}/CB_s$ blocks.
3. (At the origin server) The origin server distributes the blocks to all n edge servers in parallel; specifically, send Group k blocks to edge server k at data rate $B_{s,k}$, for $k = 1, \ldots n$.
4. (At each edge server) While receiving from the origin server, each edge server k replicates its received blocks (from the origin server) and distributes them to the remaining (n-1) edge servers at data rate $B_k/(n-1)$ each.

Fig. 3 ServerCast Scheme

individual edge servers. When blocks from the origin server are received, each edge server replicates and distributes those received blocks to the other (n-1) edge servers, at data rate $B_k/(n-1)$.

Fig. 3 presents the pseudo code for the proposed ServerCast scheme.

V. Performance Evaluation

We compare the performance of our proposed ServerCast with FastReplica [4] and Multiple Unicast using ns-2 simulator. (Multiple Unicast establishes n parallel TCP connections, one to each edge server. There is no inter-edge server communication.) We implement all the three schemes on top of TCP and a TCP segment size of 1KB is assumed. This is because content distribution requires reliable delivery service. If UDP is used, the performance of our ServerCast scheme should perform even better. This can be explained by the fact that using UDP, bandwidth allocation at application layer can be more effectively implemented (not affected by TCP's congestion control mechanisms). Also, UDP packet has a smaller transport layer header, which results in higher transmission efficiency.

We first consider a simple CDN network as shown in Fig. 4. It consists of three edge servers, with outbound bandwidths B_1=20Mbps, B_2=40Mbps, and B_3=60Mbps respectively. The round trip time among all server pairs is set to 100ms. The size of each segmented block is C=256KB, and an 800MB file is to be distributed to the three edge servers.

Assume the outbound bandwidth of the origin server is B_s=75Mbps. From Theorem 1, the optimized bandwidth allocation is found to be $B_{s,1}$=15Mbps, $B_{s,2}$=25Mbps, $B_{s,3}$=35Mbps. In our simulations, this bandwidth allocation is enforced by restricting the TCP sender's transmission window size. For FastReplica and Multiple Unicast, although there is no explicit bandwidth allocation scheme, the 75Mbps outbound bandwidth tends to be equally allocated to the three edge servers.

Fig. 5 shows the content update time against the amount of data received by the slowest edge server. We can see that ServerCast gives the lowest content update time and its performance is quite close to the theoretical lower bound (as obtained from Theorem 1). The performance gap is due to the dynamics of TCP (such as slow-start), finite block size, TCP and IP packet headers, etc. As expected, Multiple Unicast provides the poorest performance. For FastReplica, its poorer-than-ServerCast's performance is mainly due to its inability to fully utilize the available outbound bandwidths.

Figs. 6 and 7 show the downloading progress at the three edge servers using ServerCast, but with *two* different bandwidth allocations. Fig. 6 uses equal bandwidth allocation, i.e. $B_{s,1}$=$B_{s,2}$=$B_{s,3}$=25Mbps, and Fig. 7 uses the optimized bandwidth of $B_{s,1}$=15Mbps, $B_{s,2}$=25Mbps, $B_{s,3}$=35Mbps. From Fig. 6, we observe that equal bandwidth allocation results in different downloading data rates (i.e. different line slopes) for

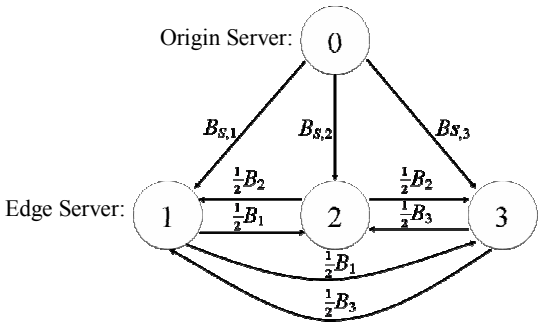

Fig. 4 A simple CDN Network

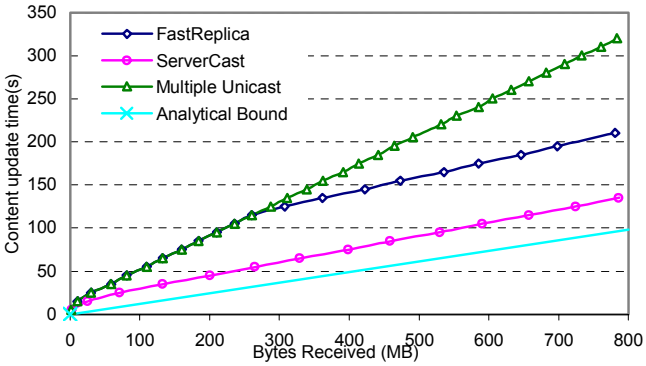

Fig. 5 Content update time with three bulk data distribution schemes: ServerCast, FastReplica, and Multiple Unicast

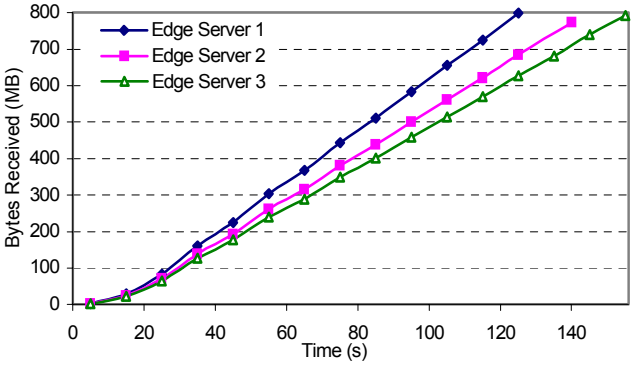

Fig. 6 ServerCast with equal bandwidth allocation; $B_{s,1}$=$B_{s,2}$=$B_{s,3}$=25Mbps

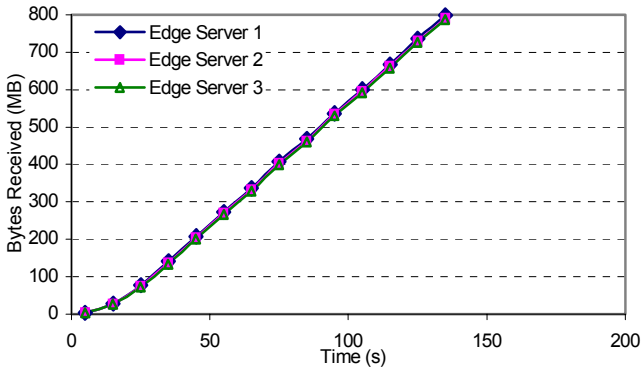

Fig. 7 ServerCast with optimized bandwidth allocation; $B_{s,1}$=15Mbps, $B_{s,2}$=25Mbps, $B_{s,3}$=35Mbps

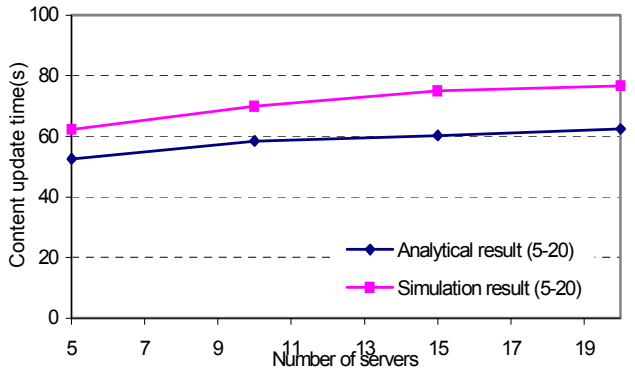

Fig. 8 ServerCast versus analytical lower bound on randomly generated topologies

different edge servers. Unavoidably, this lengthens the content update time, which is governed by the slowest downloader. From Fig. 7, we can see that the three edge servers receive the file at almost the same time. In other words, the total inbound bandwidths allocated as well as received by different edge servers are well-balanced.

Next we examine the performance of ServerCast on randomly generated topologies. In particular, the number of servers is varied from 5, 10, 15 to 20. The outbound bandwidth for each server is randomly selected from 5Mbps to 20Mbps. The round trip time among all server pairs is set to 100ms. For each generated topology, the server that has the highest outbound bandwidth is selected as the origin server. Then a 100MB file is distributed from the origin server to the remaining edge servers using ServerCast.

Fig. 8 shows the average content update time against the number of servers. Each point in Fig. 8 represents the average over 100 randomly generated topologies. We can see that the content update time of ServerCast deviates above the lower bound by 10% to 18%. As mention before, this is mainly due to the influence of congestion control mechanisms of TCP, which makes the enforcement of bandwidth allocation by application layer difficult.

VI. Conclusion and Future Work

In this paper, we studied bulk data distribution schemes for content distribution networks (CDNs) using application-level overlay. Given the outbound bandwidth of all the CDN edge servers, a fluid-flow based analytical model was constructed to derive the optimal bandwidth allocations at the origin server and among all the edge servers. A lower bound on content update time was also derived. To have an efficient and practical scheme for bulk data distribution, ServerCast was then designed based on the analytical model.

Bulk data transfer usually carries multimedia data. One possible future work is to use application-level overlay to support real-time multimedia streaming in the Internet. The time-sensitive nature of the multimedia data makes this a challenging task.

References

[1] "Akamai Technologies, Inc." http://www.akamai.com/

[2] Dinesh C. Verma., *Content Distribution Networks : An Engineering Approach*, 2002

[3] Xinyan Zhang, Gang Song, Qian Zhang and Wenwu Zhu, T.S.P. Yum, "MultiServ: congestion alleviation using overlay network" in *Proceedings of IEEE GLOBECOM 2003*

[4] Ludmila Cherkasova, Jangwon Lee, "*FastReplica*: Efficient Large File Distribution within Content Delivery Networks", *4th Usenix Symposium on Internet Technologies and Systems (USITS 2003)*

[5] R. Sherwood, R. Braud and B. Bhattacharjee, "*Slurpie*: A cooperative bulk data transfer protocol" in *Proceedings of IEEE INFOCOM 2004*

[6] B. Cohen, "Incentives Build Robustness in *BitTorrent*" in *P2P Economics Workshop, 2003*

[7] E. Cronin, S. Jamin, Cheng Jin, A.R. Kurc, D. Raz and Y. Shavitt, "Constrained mirror placement on the Internet", *IEEE Journal on Selected Areas in Communications*, vol 20, number 7, pp. 1369 – 1382, September 2002

[8] Lili Qiu, V.N. Padmanabhan and G.M. Voelker, "On the placement of Web server replicas" in *Proceedings of IEEE INFOCOM 2001*

[9] Yan Chen; Lili Qiu; Weiyu Chen; Luan Nguyen; R.H. Katz, "Efficient and adaptive Web replication using content clustering", *IEEE Journal on Selected Areas in Communications*, vol 21, number 6, pp. 979 – 994, August 2003

[10] S. Ratnasamy, M. Handley, R. Karp and S. Shenker, "Topologically-aware overlay construction and server selection" in *Proceedings of IEEE INFOCOM 2002*

[11] Adrian J. Cahill and Cormac J. Screenan, "An effcient CDN placement algortithm for the delivery of high quality TV content", in *Proceedings of ACM Multimedia 2004*

[12] N. Hu and P. Steenkiste, "Evaluation and Characterization of Available Bandwidth Probing Techniques", *IEEE Journal on Selected Areas in Communications*, Vol. 21, number 6, Aug. 2003

[13] R.S. Prasad, M. Murray, C. Dovrolis, K. Claffy, "Bandwidth estimation: metrics, measurement techniques, and tools", *IEEE Network Magazine 2003*